VZ-LINK NEWLETER

<< AUCKLAND VZ 300/200 USERS CLUB >>

ISSUE # 17 FEBRUARY 1988

The workshop will be held on 27Th February 1988 .

(Details enclosed),I hope most of you can attend.
If you can bring your VZ or some programs for us to look at,it will be a great day.

This issue see,s the second part of the assembly item .
I would like to take the opportunity to thank Mark Harwood for giving permission for us to reprint this item from his VZ USER Newsletter.

We also have the last part of Robert Quinns UYVZ item.
Luckly for us users,This is not the last you will be hearing of Robert as he has written many other fine items for the VZ.

I have also included the second part of David boyces Speech Synthesiser notes as promised.I know many of you are interested in this subject.
More in the next issue.

I am thinking of increasing the membership sub to $15-00 per 12 issue's ,if you feel that this is to expensive or the newsletter is not worth it,Please let me know.
Remember this is your club so have your say.

Sometimes in the newsletter I forget to include addresses for the people who submit programs and item,s,If you wish to make contact just give me a call.I will try to correct this bad habit.

If you are after plug's or sockets try connector system,s in khyber pass road Auckland.

Remember that any local item,s for use in the newsletter would be really great,so send them in folk,s.

So once again,until next issue or until I see you at the workshop.--HAVE FUN WITH YOUR VZ.

UNDERSTANDING YOUR VZ - (C)

NOTE :- See issue 15, page 9 for LLISTing of DLINE.

DLINE is the deluxe version of LINE. When you have entered DLINE, check the accuracy of the DATA statements with this direct command:

```
FORR=29184 TO 29260:READA:Z=Z+A:NEXT:PRINTZ
```

If your answer is not 6319, then recheck all the numbers in the DATA statements at the end of DLINE. When RUN50000, DLINE will ask for a line number. Enter the number of the program line you want analysed, or enter a basic word. If you enter a word, DLINE will search through the word table (token list table) for the word and print the token code for that word, then ask for a line number again.

When a line number has been entered, DLINE will ask for a token. If you want DLINE to seek out a particular token code in the line, then enter that token and DLINE will display each byte in the line in continuous sequence, pausing only when it encounters an instance of the token. If you do not want DLINE to search for a particular token, then simply press <RETURN> and DLINE will display the first five bytes of the line and pause. Press <SPACE> to single-step through the remainder of the line. DLINE will display the address of the memory cell for each byte, the position of the byte in the line (first,second,third,etc.), the byte itself, then the character or basic word designated by the byte (under the ? heading) and, finally, the negative PEEK/POKE address of the byte if the address is greater than 32767. Wheneyer DLINE pauses, you can press <O> or <L> or <X> key to activate a particular POKE facility.

<O>: DLINE will ask for a POKE address and then POKE the number 140 (token code for LET) to that address.

<L>: DLINE will ask for a POKE address, then for a TOKEN, and POKE the token number to the address.

<X>: DLINE will ask for a TOKEN and then POKE the token number to the last address displayed on the screen.

<6>: Pressing this key will reRUN DLINE.

```
50300 IFZ=192THEN PRINT"VARPTR";:RETURN
```

This line in DLINE recognises the token for VARPTR and lists this word when it encounters the token. Other similar lines can be inserted into DLINE for other words that are not functional in the word table. They should have line numbers greater than 50300 and less than 50320.

```
EG. 50302IFZ=196THEN PRINT"STRING$";:RETURN
```

Such lines should be inserted after DLINE has been merged with another program, because DLINE is already close to the maximum length of a program that can be merged using the MERGE routine accompanying this article. Any further lines will take it over the limit. Those of you who have a disk drive can run DLINE and then BSAVE a binary program of the machine code routines used by DLINE:-

BSAVE"LCODE",7200,724C and then delete all the DATA statements and the READ/POKE routine from DLINE. The memory gained can then be used to install a number of statements for recognising and listing various nonfunctional words as indicated above.

Line 50010 should now read:

50010 IFPEEK(29223)=33ANDPEEK(29224)= 233THEN50030 and a line 50020 entered thus:

50020 INPUTB:BLOAD"LCODE"

The INPUT statement is simply a precaution to prevent you from initiating BLOADing of LCODE without a disk in the drive or the door closed (you've had that experience too have you?).

Incidentally, if you want to use VZ disk commands in conditional statements, the trick is to insert a colon between THEN and the disk command, like this:-

80 IFK=9THEN:BLOAD"PROGRAM"

The deluxe version of line is obviously too long to type in every time you want to append it to a program. Salvation is at hand in the form of a merge routine. You need only type DLINE in once, save it on tape or disk, and thenceforth use the MER program to join DLINE to other programs.

MER allows you to join one program onto the end of another so that they become a single program. Type in MER and save. Check the length of MER with this length command:-

PRINTPEEK(30969)+PEEK(30970)*25 6-31465

The answer should be 370. Then check the machine code of the MERGE routine with this command sequence:

FORR=31273TO31354:READA:B=B+A:N EXT:PRINTB

The answer should be 9118. If not, check out all the numbers in the DATA statements. When RUN, MER sets up a machine code MERGE routine in the communications region. MERGE is the program which does the adding on.

PROG 1 is the program you wish to add on.
PROG 2 is the program to which you want to add PROG 1.

The maximum length for PROG 1 is 1534 bytes. If you have any doubts as to whether PROG 1 exceeds the maximum length, then check the length with the above command. For trouble-free merging, ensure that the line numbers for all the lines in PROG 1 are greater than all the line numbers in PROG 2.

Enter the following sequence of commands to carry out a merge (CRUN and CLOAD if you are using a cassette recorder):

```
RUN"MER"       'Transfers M. Code MERGE to communications region
LOAD"PROG 1"   'Use your name for PROG 1
PRINTUSR(8)    'MERGE copies PROG 1 to video memory
LOAD"PROG 2"   'Use your name for PROG 2
PRINTUSR(8)    'MERGE adds PROG 1 onto end of PROG 2
```

UNDERSTANDING YOUR VZ CONT. .

Most routines you would want to merge with another program would be less than the maximum length permitted (1534).

However, should you want to add on a longer routine, you can do it in stages. When you load PROG 1, chop PROG 1 in half by using the DELETE command (see Table X) to remove the top end of PROG 1, and complete the merge. Save the result; this will be your new PROG 2. Now repeat the merge sequence of commands from LOAD"PROG 1" onwards.

This time DELETE the bottom part of PROG 1 and merge the remainder with the new PROG 2.

Once MER has been run the MERGE routine can be used to merge any number of pairs of programs without having to reRUN the MER program each time.

Merge will add on PROG 1 to PROG 2 regardless of the line numbers of the two programs. The result could be that a routine is joined to the end of PROG 2 that has the same line numbers as PROG 2 or some part of PROG 2. This is an opportunity to get two or more programs to reside in program memory simultaneously. With the proper POKEs to start of program pointer you can then RUN the program of your choice.

LISTing for MERge :-

```
10 FORR=31273TO31354:READA:POKER,A:NEXT
20 POKE30862,41:POKE30863,122
30 DATA175,42,249,120,17,233,122,237,82,68,77,237,67,0,114,33
40 DATA233,122,17,2,114,237,176,33,71,122,34,142,120,201,237
50 DATA91,249,120,27,27,107,98,237,75,0,114,9,34,249,120
60 DATA33,2,114,237,176,33,41,122,34,142,120,175,17,233,122,98
70 DATA107,190,35,32,2,190,200,35,35,35,190,32,252,35,235,115,35
80 DATA114,24,235
```

[C] COPYRIGHT is retained by ROBERT QUINN on the series UNDERSTANDING YOUR VZ.

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME BETWEEN 6.30 PM AND 9 PM ----PHONE 673-967

EXT. DOS V1.0 — BRIEF REVIEW

EXTENDED DOS VERSION 1.0 (C) - COPYRIGHT - DAVE MITCHELL - 1987

COMMANDS :-

MERGE - MERGES basic file from disk with program in memory.

DIRA - See example - T:MENU B:PATCH3.1 B:WORDPROC
B:EXTDOS E B:EXTDOS R W:DOS-INST
LDIRA - As above, but to screen and printer.

DIRB - See example - T:MENU 01 00 7AE9 801B 0532
B:PATCH3.1 01 0B 7200 771F 051F
LDIRB - As above, but to screen and printer.

STATUSA - Prints free disk space to screen on one line.
LSTATUSA - As above, but to screen and printer, see below.
534 RECORDS FREE 63.500K FREE

OLD - Restores a program after using the NEW command.

OLD. - Prints START, END and LENGHT of program in memory in HEX.

DEC XXXXX - Converts DECIMAL to HEX

HEX XXXX - Converts HEX to DECIMAL

STATUSA and LSTATUSA also works with Version 1.0 DOS.

The EXTENDED DOS is available in the two versions below :-

EXTDOS R - T.O.M. SEEKING
EXTDOS E - FOR 2K RAM AT 6000-67FF HEX

Price - $10.00 each or the two for $15.00. Availble from :-

Dave MITCHELL - (079) 27 8519 24 ELPHINSTONE STREET
NORTH ROCKHAMPTON QUEENSLAND 4701

NEWCASTLE AREA :- Joe LEON - (049) 51 2756
22 DRURY STREET WALLSEND N.S.W. 2287

BRIEF USER REPORT by Joe LEON :-

The above utilities work with both V1.0 and V1.2 DOS. At long last V1.0 DOS owners can have a STATUS command. The best thing about EXTDOS R/E is they extend the operating system and if you have a 2K ZERO POWER RAM at 6000-67FF HEX like I have then all above commands are available at power up.

EXTDOS R/E is one of these utilities that once used you don't want to be without. Like any utility it can be improved, certainly not the existing commands, but by adding more. Dave Mitchell sure has done a nice job on it.

DIRA and DIRB are my favourites. The first one displays over 45 files on the screen while the second one gives lots of information on your disk files.

Z80 ASSEMBLY

Last month we started the series with a look at the fundamental loads that are available on the Z80. This month we'll be looking at several other commands, and at a small simple program in Assembly. For those of you who have purchased the Editor/Assembler we will also go through the steps in writing a program in Assembly using it.

We said earlier that in some ways Assembly is similar to BASIC. Instructions are given a line each, and the CPU executes them one after another. As we're about to see, some assembly instructions are very similar to BASIC commands in what they do.

One essential function of any programming language is that it provides the facility to form 'loops', in order that the program can carry out repeditive functions endlessly if necessary. BASIC provides such commands as GOTO, GOSUB, FOR-NEXT. Z80 assembly has equivalents to just about all of these. Let's look at them now.

The Z80 equivalent to GOTO is JP (stands for jump), and JR (stands for jump relative). Firstly, we'll consider JP. JP simply tells the Z80 to jump to another place in memory, and start executing from the new address. It differs from GOTO in that it does not use line numbers to specify where the Z80 must jump to. The jump address must be a number representing the address itself. Using the Ed.Asm. the address can be written in Hex or decimal, i.e.:

JP 36864 and JP 9000H

both tell the Z80 to jump to 36864, and start executing the instruction found there. Using the Ed.Asm. another version of the JP instruction is offered. Last month we spoke of labels, where a line is given a name if you like, a four letter label at its start. Whenever the Ed.Asm. comes across one of these labels while assembling your program, it records the address at which the label would have appeared in the object code. This probably sounds confusing, so let's look at an example:

Suppose this was the first line of your program:

```
001 STRT   LD HL,4000H
```

and you set the origin (start address) of the program at 31465, then the Ed.Asm. would remember that STRT was the same as saying '31465'! Now, if in your program you wanted the Z80 to jump back to the start of your program at some point, then this could be done in several ways. Firstly, knowing that you set the origin to 31465 (7AE9H) you could jump to the start with:

JP 31465 or JP 7AE9H

However, what if you were still in the process of writing the program, and you didn't know what the origin was to be? You could have used:

JP STRT

In this case, the Ed.Asm., seeing 'STRT' would immediately know where you wanted to jump to. It would fill in the jump address as 31465 automatically in your object code! You can imagine how useful this option is when you need to jump to some spot deep in a program, whose address you'd have no idea of! If you're interested in how the CPU actually jumps, here's a quick explanation. The Z80, after receiving the JP code of 195 (C3H) simply loads the next two bytes of the program, which are a Hi-Lo byte pair pointing to the jump address, into the Program Counter.

Other versions of the JP instruction exist, but we'll be looking at these later. Now let's look at the JR instruction. JR (jump relative) achieves the same result as JP, the only difference is the way that the address to jump to is specified. With JR, the Z80 is told to jump to an address either forward in memory, or back in memory from its current position. Put simply, the address to jump to can be specified in the range of -126 to +129 bytes, meaning any address up to 129 bytes upward in memory, or up to 126 bytes downward in memory.

Although the exact offset can be specified using the Ed.Asm., because of the difficulty involved in calculating the value, JRs are used mainly with labels. But be careful when using them, as if an address is out of the range specified above, you will get a 'BRANCH OUT OF REACH' error when assembling. You may then ask what use JRs are to the VZ programmer. Their only real advantage over JPs is that they are relocatible, i.e. no matter where the program is in memory, the JR will still jump to the correct address. Also JRs use one less byte of memory, but are slightly slower than JPs.

One last point about JPs, they can be used with indirect addressing, where a register pair specifies the address to jump to. For example:

```
JP (HL)
```

tells the Z80 to jump to the address stored in HL. The HL, IX and IY registers only can be used in this fashion.

We mentioned that an assembly version of the GOSUB command exists - this is the CALL instruction. CALL does two things, firstly the Z80 jumps to the address specified, and secondly, the next address after the CALL is placed on the stack (to be discussed later). When the CPU comes across the next RET (return from subroutine) instruction, it gets this location back off the stack and continues on from there, just as BASIC does after receiving a RETURN. As with JP, CALL can be used with an absolute address, or a label. Indirect addressing with CALL is not supported on the Z80. Conditional JPs and CALLs are supported, but we'll look at these later.

Let's leave the theory and study a real program. Firstly, here's the listing of the Source Code:

```
001 LOOP   LD HL,50      ;set 1st frequency
002        LD BC,500     ;set duration
003        CALL 345CH    ;call sound routine
004        LD HL,60      ;set 2nd frequency
005        LD BC,500     ;set duration
006        CALL 345CH    ;call sound routine
007        JR LOOP
```

Now let's go through all the steps involved in getting the program going using the Ed.Asm. After loading the Ed.Asm., enter I for Insert. Line 001 should appear with a cursor. Type in the line as it appears above, remembering to leave a space between the label and op-code, and to leave out the comments. After pressing return, the next line number will appear. Now you must hit space before each line, as none of the rest of the program has labels. After hitting return for the last line, hit control-break to get back to the command mode. Next step is to set the origin. Since we don't expect to reload our program, simply type '0' and the Ed.Asm. will set its own origin. This allows us to run the program with the Ed.Asm. still in memory. Now that the source has been entered, we must assemble the object code itself. Enter 'A' for assemble, and if any errors appear, fix them up using the edit facility. (see manual). Now type 'R' for Run, and 'Y' for yes. Your VZ should produce a siren-type sound.

Now let's study the short routine in detail. First we need to know something about the sound routine, which starts at 345CH in ROM. This routine produces a tone from the piezo speaker. The HL registers must contain a pitch code, and the BC register pair a duration code. In lines 1 & 2 these are loaded respectively. Line 3 CALLs the sound routine, and after the routine is complete, control returns to line 4, where a new frequency (pitch) and duration are loaded. After the sound routine has been CALLed again, a Jump Relative is used to start the routine all over again. The program continues indefinitely as we have made no escape option.

Experiment with the routine yourself. Next month we'll look at how to insert a keyboard routine to break the program, and find out about several more Z80 op-codes, and about that thing they call the 'stack'!

TWISTER

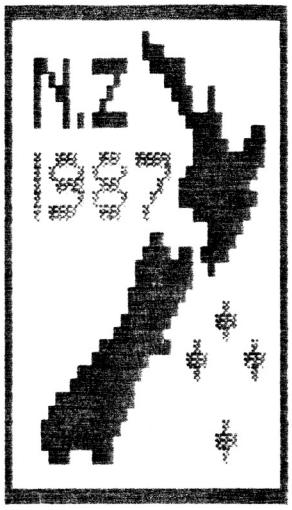
WRITTEN BY R.G DAVIS

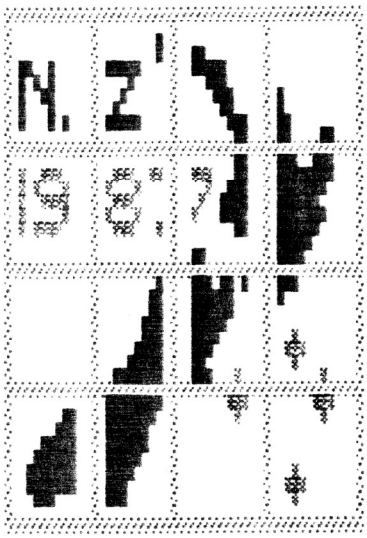
IN THIS PUZZLE GAME,YOU WILL GET A MAP OF NEW ZEALAND MADE UP
FROM SIXTEEN BLOCKS (A FOUR BY FOUR GRID).
YOU GET A LOOK AT THE PUZZLE,THEN THE COMPUTER MIXES THEM UP
YOU HAVE TO GET THEM BACK INTO PLACE USING THE KEY'S FROM 1 TO 9
IF YOU PRESS KEY ONE THE TOP LEFT FOUR SQUARES TURN CLOCKWISE,
THAT IS SQUARES 1,2,5,6.
IF YOU PRESS KEY 2 THEN YOU TURN SQUARES 2,3,6,7
IF YOU PRESS KEY 5 YOU TURN SQUARES 6,7,10,11 AND SO ON
UNTIL YOU HAVE THEM ALL BACK INTO PLACE.
THERE IS ALSO AN EXPLODED VIEW OF THE PICTURE AS IT WAS ALMOST
IMPOSSIBLE TO REMEMBER WHERE THEY ALL BELONGED
ALSO THERE IS A HELP KEY IF YOU CAN'T GET IT ALL TOGETHER PRESS 'H'
YOU CAN PRESS ANY KEY AT THE START TO MIX THE SQUARES UP

```
1 CLS:GOSUB 500
2 CLEAR 500:DIM B(16)
5 MODE(1):COLOR4,1:X=14:FOR N=8TO 46:SET(N,X):SET(N,X+36):NEXT
7 X=8:FOR N=14TO50:SET(X,N):SET(X+38,N):NEXT
8 GOSUB 308
9 H=-12794:L=-12795:Q=-12755:FOR N=1 TO 16:B(N)=N:NEXT
10 GOSUB 110:GOSUB 210:FOR N=1 TO 500:NEXT
11 K$=INKEY$:K$=INKEY$:IF K$=""THEN 11
12 GOSUB 80
14 K$=INKEY$:K$=INKEY$:IF K$="" THEN 14
15 IF K$="H"THEN 9
16 X=VAL(K$):IF X=0 THEN 14
18 IF X<4 THEN X=X+1:GOTO 24
20 IF X<7 THEN X=X+2:GOTO 24
22 X=X+3
24 GOSUB 86:GOTO14
80 T=1
82 X=RND(12)
84 IF X=1 OR X=5 OR X=9 THEN 82
86 A=B(X):B(X)=B(X-1):B(X-1)=B(X+3):B(X+3)=B(X+4):B(X+4)=A
90 GOSUB 110
92 T=T+1:IF T<10 THEN 82 ELSE RETURN
100 GOTO 100
110 POKE Q,B(1):POKEH,114:POKEL,3:GOSUB 200
115 POKE Q,B(2):POKEL,5:GOSUB 200
120 POKE Q,B(3):POKEL,7:GOSUB 200:POKEQ,B(4):POKEL,9:GOSUB 200
125 POKEQ,B(5):POKEH,115:POKEL,3:GOSUB 200:POKEQ,B(6):POKEL,5
130 GOSUB200:POKEQ,B(7):POKEL,7:GOSUB200:POKEQ,B(8):POKEL,9
132 GOSUB 200
```

```
135 POKEQ,B(9):POKEH,116:POKEL,3:GOSUB200:POKEQ,B(10):POKEL,5
137 GOSUB 200
140 POKEQ,B(11):POKEL,7:GOSUB200:POKEQ,B(12):POKEL,9:GOSUB200
145 POKEQ,B(13):POKEH,117:POKEL,3:GOSUB 200
150 POKEQ,B(14):POKEL,5:GOSUB200:POKEQ,B(15):POKEL,7:GOSUB200
155 POKEQ,B(16):POKEL,9:GOSUB 200
160 RETURN
200 X=USR(0):RETURN
210 REM
211 POKEQ,B(1):POKEH,114:POKEL,16:GOSUB 200
212 POKE Q,B(2):POKE L,19:GOSUB 200:POKE Q,B(3):POKE L,22
214 GOSUB 200:POKE Q,B(4):POKE L,25:GOSUB 200
216 POKE Q,B(5):POKE H,115:POKE L,48:GOSUB 200
218 POKE Q,B(6):POKE L,51:GOSUB 200
220 POKE Q,B(7):POKE L,54:GOSUB 200
222 POKE Q,B(8):POKE L,57:GOSUB 200
224 POKE Q,B(9):POKE H,116:POKE L,80:GOSUB 200
226 POKE Q,B(10):POKE L,83:GOSUB 200
228 POKE Q,B(11):POKE L,86:GOSUB 200
230 POKE Q,B(12):POKE L,89:GOSUB 200
232 POKE Q,B(13):POKE H,117:POKE L,112:GOSUB 200
234 POKE Q,B(14):POKE L,115:GOSUB 200
236 POKE Q,B(15):POKE L,118:GOSUB 200
238 POKE Q,B(16):POKE L,121:GOSUB 200
300 RETURN
308 COLOR 2:X=14:FOR N=62 TO 110:SET(N,X):SET(N,X+38)
310 SET(N,X+10):SET(N,X+19):SET(N,X+28):NEXT
320 X=62:FOR N=14 TO 52:SET(X,N):SET(X+49,N):SET(X+12,N)
322 SET(X+23,N):SET(X+36,N):NEXT
330 RETURN
400 DATA 245,197,213,229,33,121,117,205,41,206,14,8,26,119
402 DATA 35,19,26,119,6,31,35,5,120,254,0,194,20,206,19,13,121
404 DATA 254,0,194,12,206,241,193,209,225,201,17,43,206,14,16
406 DATA 6,16,19,16,253,13,121,254,0,194,46,206,201,0,0,0,0
408 DATA 240,192,204,192,195,192,192,192,192,204,0,0,0,3,0,3
410 DATA 255,192,3,0,12,0,48,0,255,192,0,0,0,0,240,0,255,192
412 DATA 15,192,3,240,0,240,0,63,0,15,0,0,0,0,0,0,0,192,0,192
414 DATA 0,240,0,240,15,0,0,130,160,136,8,130,168,128,8,138
416 DATA 160,0,0,0,0,0,0,42,10,128,128,42,0,128,128,42,2,0,0,0
418 DATA 0,0,15,168,15,8,63,32,255,128,255,0,15,0,0,240,0,252
420 DATA 63,255,255,255,252,255,240,255,252,255,240,255,192
422 DATA 255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,15,0,15,0
424 DATA 63,0,63,0,255,3,255,63,255,252,243,255,240,255,240
426 DATA 255,192,255,0,252,0,240,0,252,8,252,0,192,0,0,0,0,0,8
428 DATA 0,42,0,8,0,0,8,0,0,0,15,0,63,3,255,15,255,63,255,63
430 DATA 255,15,192,255,255,255,252,255,240,255,240,255,192
432 DATA 255,0,252,0,252,0,0,42,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0
434 DATA 42,0,8,0,0,0,0,0,0,8,0,42,0,8,0
500 POKE 30897,255:POKE 30898,205
505 FOR N=-12800 TO -12486:READ M:POKE N,M:NEXT
510 POKE 30862,0:POKE 30863,206
520 RETURN
```

SCREEN DUMP OF TWISTER SCREEN
USING LARRY TAYLORS PRINTER PATCH V1.4





LOAN

WRITTEN BY K.WRIGHT

THIS PROGRAM WILL HELP YOU
WORK OUT YOUR LOAN/MORTGAGE REPAYMENTS

```
50 CLS
60 PRINT @ 330,"W A R N I N G ! !"
70 PRINT TAB(6);"▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼"
80 PRINT:PRINT" THIS PROGRAM COULD BE HARMFULL"
90 PRINT"TO YOUR MORALE AND MENTAL HEALTH"
100 FOR ZZ=1 TO 2157:NEXT
110 CLS:PRINT@260,"MORTGAGE REPAYMENT"
120 FORN=31TO12STEP-1:SOUNDN,INT(4-N/13):NEXT:SOUND1,7
130 CLS:PRINT:U$="$$########,.##"
140 INPUT"    PRINCIPAL";P
150 INPUT"          TIME";T
160 INPUT"INTEREST RATE";R:R=R/100
170 REP=((1+R)^T)*R*P/(((1+R)^T)-1)
180 PRINT:PRINT" ANNUAL PAYMENT =";
190 PRINTUSINGU$;INT(REP*100)/100
200 PRINT"MONTHLY PAYMENT =";
210 PRINTUSINGU$;INT(REP*8.33)/100
220 PRINT" WEEKLY PAYMENT =";
230 PRINTUSINGU$;INT(REP*1.923)/100
240 PRINT" TOTAL PAYMENTS =";
250 PRINTUSINGU$;INT(REP*T*100)/100
260 PRINT@448,"->->[SPACE] FOR NEXT INPUT->"
270 PRINT" ->-> OR [Q] TO QUIT ->->";
280 IF INKEY$<>""THEN GOTO280
290 K$=INKEY$:K$=INKEY$:IF K$="" THEN GOTO290
300 IF INKEY$<>""THEN GOTO300
310 IF K$=" " THEN RUN 130
320 IF K$="Q" THEN:RUN"MENU"
330 SOUND3,2;2,3:GOTO290
```

AEM4505 TONE CONTROL BY D.BOYCE

Now that You have made Your AEM4505 SPEECH SYNTHESISER, let's add what could be best described as a TONE control. Before continuing with the construction changes, a word of WARNING.

The two Tone controls should be left in their central positions and only moved minimum amounts to suit ones' own taste. Move the controls TOO far and the Speech Synth will go into Oscilation, if the RESET button doesn't regain control then powering down of the Speech Synth (only) will be necaessry.

YOU have been WARNED.

Construction Notes :-

1> Using a small piece of Vero-board, construct the circuit diagram as shown. I suggest that sockets be installed under the two Ic's. Although the diagram looks involved, there are very few wires to connect to the Main 4505 P.C.B. These being :-

GND - 0 Volts
+ 5 Volts
1 wire from the output of the 74LS00 gate to the SPO chip plus
2 wires to the PITCH control and
3 wires to the RANGE control from the small board to the pots.

2> Remove capacitors C4 and C5.

3> Remove X-TAL2 , it's next to the SPO chip.

Reconnecting Notes :-

4> Solder the GND (0 V) wire to the 4505 P.C.B. followed by the +5 Volt wire.

5> Next - solder the wire marked TP on the diagram below into the hole nearest the SPO chip left vacant by the X-TAL. This hole is connected on the P.C.Board to pin 27 of the SPO chip.

6> Connect the remaining wires to their respective control pots.

7> On the BACK of each Control Pot, assuming they are metal, scratch the surface until you have a clean surface. Tin each of these bare spots. From the GND terminal of either the VOL or RANGE pot, run a wire to the back of each pot and solder to the 'Tinned Bare Spots'. This will stop (hopefully) the pots acting as antennas when you touch them - honest.

8> Reconnect any other wires that may have been disconnected when you started on the job.

9> Set the two new pots to their central positions.

O.K. Connect the Speech Synth back onto the VZ and turn on. Everything alright ? Did you get the familiar ' OK ' from the Speech Synth ? If not then turn off and check everything from the start.

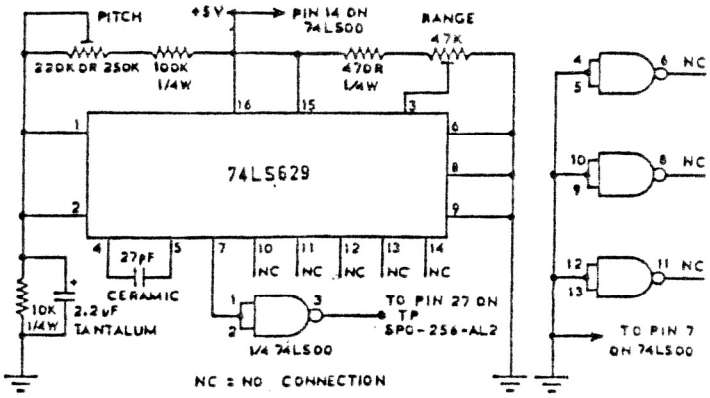
Load in and RUN "TALK A" and check that the two controls are working.

NOTE :- Circuit Diagram and Parts List on next page . . .

AEM4505 TONE CONTROL CONT. . .

Parts List for the Tone Control for AEM 4505 Speech Synth. Dick Smith catalogue numbers given where possible.

1 - 74LS629 - ALTRONICS Z 8429
1 - 74LS00 - DSE Z-4900
1 - 470R resistor - DSE R-0566
1 - 10K resistor - DSE R-0598
1 - 100K resistor - DSE R-0624
1 - 27pF ceramic capacitor - DSE R-2245
See note below.
1 - 2u2 35 volt Tantalum capacitor - DSE R-4730
1 - 47K or 50K potentiometer - DSE R-1809
1 - 220K potentiometer. Yes Virginia , they do exist.
1 - 14 PIN DIP socket - DSE P-4140
1 - 16 PIN DIP socket - DSE P-4160



Note on 27pf cap.:- The value of this cap. may have to be varied from anywhere between 18pf and 33pf. A trimmer cap. from D.Smiths can also be used in place of the fixed value cap. D.S. No# - R2930. It is possible to mount the small board on the back of one of the control pots, this will save using long wires to that pot, PLUS, it saves having to find a mounting place for the small board.

I have made this modification to both my original Speech Synth and my (home made version of the) AEM4505, in either case, the mod works quite well.

Cost - $12 to $15 - Happy Hackin'. Dave Boyce June '87

HAM CONTACT LOG

FROM MICRO MAGIC BY GORDON BROWELL

```
10 '***************   AMATEUR
20 '* MICRO MAGIC * CONTACT LOG
30 '***************   27APR86
40 'SEARCH LISTS FROM ANY DATA.
50 'ABC WOULD BE SUFFICIENT TO
60 'FIND C/S VK4ABC.
70 'THE VZ ADDS Z TO TIMES AND
80 'M FOR METRES.
100 CLEAR8000:DIMZ$(200):CLS
110 X$="VK4AAA":Y$="27 APRIL 1986"
120 PRINT@100,"ENTER CALLSIGN";:INPUTX$
130 PRINT@164,"ENTER DATE";:INPUTY$:CLS
140 Z$(0)=X$+" CONTACT LOG "+"                "+Y$
149 '"MENU
150 CLS:PRINT@34,Z$(A)
160 PRINT@34,Z$(A)
170 PRINT@130,"ENTER    LIST   FILE   MEND"
180 PRINT@194,"SEARCH   PRINT  RETRIEVE"
190 PRINT@258,"ENTER INITIAL LETTER"
200 M$=INKEY$:M$=""
210 M$=INKEY$:IFM$=""THEN210
```

```
220 IFM$="E"THEN290
230 IFM$="S"THEN380
240 IFM$="L"THEN480
250 IFM$="P"THEN490
260 IFM$="F"THEN500
270 IFM$="A"THEN610
280 IFM$="R"THEN540ELSE160
289 '"ENTER DATA
290 CLS:A=A+1
300 PRINT@33,Z$(A-1):PRINT@132,"TIME     ";:INPUTA$:A$=A$+"Z"
310 PRINT@196,"CALLSIGN";:INPUTB$:IFLEN(B$)=5THENB$=B$+" "
320 PRINT@260,"WAVEBAND";:INPUTC$:C$=C$+"M"
330 PRINT@324,"RX FREQ ";:INPUTD$
340 PRINT@388,"TX FREQ ";:INPUTE$
350 Z$(A)=A$+" "+B$+" "+C$+" "+D$+" "+E$:CLS
360 PRINT@100,"CONTACT LOGGED":PRINT:PRINT" ";Z$(A)
370 GOSUB630:GOTO150
379 '"SEARCH
380 Q=0
390 PRINT@258,"ENTER INFORMATION     ":PRINT
400 PRINT" ";:INPUTD$:D=LEN(D$):CLS
410 FORI=1TOA:B=LEN(Z$(I))
420 FORJ=1TOB:
430 S$=MID$(Z$(I),J,D)
440 IFS$=D$THENPRINTI;Z$(I):Q=1
450 NEXT:NEXT:PRINT"          SEARCH ENDS"
460 IFQ=0THENPRINT"           NOT FOUND":Q=0
470 GOSUB630:GOTO640
480 CLS:FORI=1TOA:PRINTI:PRINT" ";Z$(I):NEXT:PRINT:GOTO640
489 '"PRINTER
490 FORI=0TOA:LPRINTZ$(I):NEXT:GOTO160
499 '"TAPE FILE
500 CLS:PRINT@164,"STANDBY TO RECORD"
510 GOSUB630:PRINT@255,"START RECORDING"
520 FORI=0TOA:PRINT#"",Z$(I):PRINTZ$(I):NEXT
530 PRINT"STOP TAPE":GOSUB630:GOTO150
539 '"RETRIEVE
540 CLS:PRINT@100,"RETRIEVE"
550 PRINT:PRINT"ENTER HOW MANY ENTRIES";:INPUTA:CLS
560 FORI=0TOA:INPUT#"",Z$(I):PRINTZ$(I):NEXT:PRINT"STOP TAPE"
570 GOSUB630:GOTO150
579 '"EDIT
580 CLS:PRINT@99,Z$(A)
590 PRINT@64,"":INPUTZ$(A)
600 GOTO150
609 '"AMEND
610 PRINT@258,"ENTER  LINE NUMBER";:INPUTF:PRINT@323,Z$(F)
620 PRINT@288,"":INPUTZ$(F):GOTO150
629 '"DELAY LOOP
630 FORI=1TO2000:NEXT:RETURN
639 '"CONTINUE
640 'M$=INKEY$:M$=""
650 PRINT@418,"CONTINUE MEND SEARCH LIST":GOTO200
660 'M$=INKEY$:IFM$=""THEN660
670 'IFM$="C"THEN150ELSE660
```

DON'T FORGET THE MEETING/WORKSHOP
27TH FEBRUARY 1988